Contextual graph grammars characterizing context-sensitive languages

Christophe Morvan

Publications Internes de l'IRISA
ISSN : en cours
PI 1926 – Mars 2009

Contextual graph grammars characterizing context-sensitive languages*

Christophe Morvan**
christophe.morvan@irisa.fr

Abstract: Deterministic graph grammars generate a family of infinite graphs which characterize context-free (word) languages. In this paper we presents a context-sensitive extension of these grammars. We achieve a characterization of context-sensitive (word) languages. We show that this characterization is not straightforward and that unless having some rigorous restrictions, contextual graph grammars generate non-recursive graphs.

Key-words: Graph grammars, Context-sensitive languages, Contextual graph grammars

---

Des grammaires de graphes contextuelles qui charactérisent les langages contextuels

Résumé : *Les grammaires de graphes sont un outil permettant de définir une famille générale de graphes infinis. Les graphes en question constituent une caractérisation des langages algébriques. En elle-même la caractérisation en terme de grammaire permet de mettre en évidence la* structure *de chaque graphe.*

Une généralisation naturelle des grammaires de graphes est obtenue par les grammaires contextuelles de graphes. Jusqu'à présent ces dernières n'ont été que très peu étudiées. La principale faiblesse de ces grammaires est que, en l'absence d'une restriction, elles engendrent des familles de graphes non-récursif (autrement dit, étant donné un tel graphe, il est indécidable de savoir si un arc donné est présent dans le graphe). Cette limitation est totalement rédhibitoire.

Dans le présent rapport, on identifie un ensemble de restrictions sur les grammaires contextuelles de graphes permettant d'assurer la récursivité du graphe. Plus encore on présente une famille qui caractérise précisément les langages contextuels (il s'agit des langages engendrés par les grammaires contextuelles de mots). Ce travail présente donc une continuité directe par rapport aux grammaires de graphes ordinaires.

Mots clés : *Grammaires de graphes, Langages contextuels, Grammaires de graphes contextuelles*

---

* The present document is a second version of this research report. It has been enhanced by several figures, as well as a couple of straightforward applications.

** Vertecs - University Paris-Est

Contents

1 Introduction

In 1956, and then in 1959 Noam Chomsky wrote two articles which defined the Chomsky hierarchy. This hierarchy has had a tremendous impact on the development of the theory of formal languages. Since the early sixties, it has been a reference for the classification, and the evaluation of the expressive power of formal languages.

There is a deep connection between this hierarchy and graphs. It is obvious for type 3 languages (regular languages) which are characterized by finite automata. But from the mid eighties on, there has been an increasing effort to characterize the families of the hierarchy in terms of graphs.

For type 2 languages (context-free languages) one of the first attempt to establish properties on structures characterizing these languages is from Muller and Schupp who established the decidability of the monadic second order theory of the graphs of pushdown automata [MS85]. In their work, the vertices of the graphs are configuration of the pushdown automaton, and the arcs are transitions between configuration. That kind of characterization is called *internal*, as it thoroughly depends on the choice of the machine, and it gives an explicit name to each vertex. Courcelle in [Cou90] extended the decidability of monadic second order theory to graphs generated by deterministic graph grammars. Indeed these graph grammars were classical graph rewriting system used to characterize families of graphs. In this paper they were used to define infinite graphs. Indeed these graphs are very close to graphs of pushdown automata ([CK01]) but it is an *external* characterization: each grammar generates a set of isomorphic graphs. The name of the vertices are not stated explicitly along the generation of the graph. This is really important as it provides a higher level of abstraction. Another slight extension of these graphs is the prefix-recognizable graphs from [Cau96]. In this paper Caucal provides both an external and internal characterization of these graphs. He also proves that they have a decidable monadic second order theory, and that they characterize context-free languages.

For type 1 languages (context-sensitive languages), there are also several graph characterizations: transition graphs of linear bounded Turing machines [KP99], rational graphs [Mor00], automatic graphs [BG00, Ris03] or linear bounded graphs [CM06a]. All these work provide internal characterizations.

In this paper we propose an external characterization of context-sensitive languages, in terms of contextual rewriting system. It is organised in two parts. The first one precisely recalls the definition of regular graphs (which are generated by deterministic graph grammars), it also presents rational graphs which characterize context-sensitive languages using rational transductions. Then, the second part explores context-sensitive graphs rewriting systems. First we examine a natural, unrestricted, contextual extension of graph grammars. We show that it is too general, as it produces non-recursive graphs. Then we propose a restriction, which contains all rational graphs, and therefore context-sensitive languages. Afterwards, use a slightly more limited rewriting system which permits to establish converse inclusion. Then we show that the most obvious relaxations of this limitation yield families of non-recursive graphs.

2 Preliminaries

2.1 Mathematical notations

For any set E, its powerset is denoted by 2^E; if it is finite, its size is denoted by $|E|$. Let the set of non-negative integers be denoted by $\mathbb{N}$, and $\{1, 2, 3, \ldots, n\}$ be denoted by $[n]$. A monoid M is a set equipped with an associative operation

(denoted $\cdot$) and a (unique) neutral element (denoted ε). A monoid M is *free* if there exist a finite subset A of M such that $M = A^* := \bigcup_{n\in\mathbb{N}} A^n$ and for each $u \in M$ there exists a unique finite sequence of elements of A, $(u(i))_{i\in[n]}$, such that $u = u(1)u(2)\cdots u(n)$. Elements of a free monoid will be called words. Let u be a word in M, $|u|$ denotes the length of u and $u(i)$ denotes its ith letter.

Graphs

A (simple oriented labelled) *graph* G over V with arcs labelled in P is a subset of $V \times P \times V$. An element (s, a, t) in G is an *arc* of *source* s, *goal* t and *label* a (s and t are *vertices* of G). We denote by $Dom(G)$, $Im(G)$ and V_G the sets respectively of sources, goals and vertices of G. Each arc (s, a, t) of G is identified with the labelled transition $s \xrightarrow[G]{a} t$ or simply $s \xrightarrow{a} t$ if G is understood.

A graph G is *deterministic* if distinct arcs with same source have distinct label: $r \xrightarrow{a} s \ \wedge \ r \xrightarrow{a} t \ \Rightarrow \ s = t$. A graph is (source) *complete* if, for every label a, every vertex is source of an arc labelled a: $\forall a \in P,\ \forall s \in V_G,\ \exists t\ s \xrightarrow{a} t$. The set $2^{V\times P^+\times V}$ of graphs with vertices in V, labelled by elements of P^+, is a semigroup for the *composition relation*: $G \cdot H := \{r \xrightarrow{a\cdot b} t \mid \exists s, r \xrightarrow[G]{a} s \wedge s \xrightarrow[H]{b} t\}$ for any $G, H \subseteq V \times P^+ \times V$. The relation $\xrightarrow[G^+]{u}$ denoted by $\overset{u}{\underset{G}{\Longrightarrow}}$ or simply $\overset{u}{\Longrightarrow}$ if G is understood, is the existence of a *path* in G labelled u in P^+. For any subset L of P^+, we denote by $s \overset{L}{\Rightarrow} t$ that there exists u in L such that $s \overset{u}{\Rightarrow} t$.

A *graph morphism* g is a mapping from a graph G to a graph G' such that if there is an arc $u \xrightarrow[G]{a} v$, then there is an arc $g(u) \xrightarrow[G']{a} g(v)$.

2.2 Graph grammars

When dealing with infinite graphs, having a finite presentation is a very precious tool. It allows to manipulate or to devise algorithms for these objects. Deterministic (hyperedge replacement) graph grammars are a very nice example of finite (external) characterization of infinite graphs. These grammars were initially defined to be an extension to graphs of word grammars. Indeed such a graph grammar derived, from an axiom, an infinite family of *finite* graphs. Courcelle in [Cou90] used the deterministic form of these grammars to obtain a single infinite graph as the least solution of a finite set of deterministic graph equations. In 2007 Caucal made a very in-depth survey on deterministic graphs grammars [Cau07]. In particular he devised several techniques which allowed the presentation of these results in a very unified manner.

In order to define formally graph grammars, we recall some elements on hypergraphs. Let F be an alphabet ranked by a mapping $\rho : F \to \mathbb{N}$, this mapping associates to each element of F its *arity*. Furthermore, for a ranked alphabet F, we denote by F_n the set of symbols of arity n. Now given V an arbitrary set, a *hypergraph* G is a subset of $\cup_{n\geqslant 1} F_n V^n$. The vertex set of such a hypergraph is the set $V_G = \{v \in V \mid FV^*vV^* \cap G \neq \emptyset\}$, in our setting, this set is either finite or countable. A hyperarc of arity n is denoted by $f\ v_1\ v_2\ \cdots\ v_n$. Notice that for hyperarcs of arity 2 which are plain arcs, we will use, depending on the context, either this notation or the previous one.

Definition 2.1 (Hypergraph grammar)**.** A *hypergraph grammar* (HR-grammar for short) G, is a 4-tuple (N, T, R, H_0), where N and T are two ranked alphabets of respectively *non-terminals* and *terminals* symbols; H_0 is the axiom, a finite graph formed by hyperarcs labelled by $N \cup T$, and R is a set of rules of the form $f\ x_1\ \cdots\ x_{\varrho(f)} \to H$ where $f\ x_1\ \cdots\ x_{\varrho(f)}$ is an hyperarc joining disjoint vertices and H is a finite hypergraph.

Remark 2.2. In this paper, we consider graphs, therefore, the terminal symbols will have either rank one, or two. Furthermore, we see such a graph as a simple subset of $T_2VV \cup T_1V$. Rank 1 symbols will be called *colours* rather than labels (we use label to identify (hyper) arcs). A single vertex may have several colours.

A grammar is deterministic if there is a single rewriting rule per non-terminal:

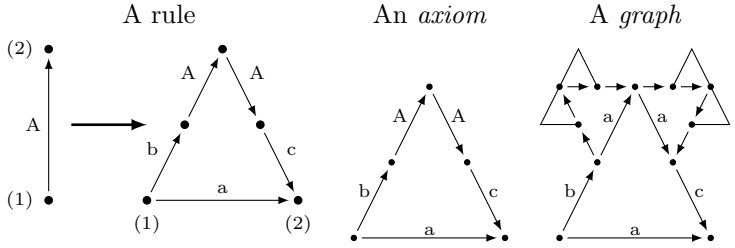
$$(X_1, H_1), (X_2, H_2) \in R \wedge X_1(1) = X_2(1) \Rightarrow (X_1, H_1) = (X_2, H_2)$$

Now, given a set of rules R, the *rewriting* $\xrightarrow[R]{}$ is the binary relation between hypergraphs defined as follows: M rewrites into N, written $M \xrightarrow[R]{} N$ if there is a non-terminal hyperarc $X = Av_1v_2\ldots v_p$ in M and a rule $Ax_1x_2\ldots x_p \to H$ in R such that N is obtained by replacing X by H in M: $N = (M - X) \cup h(H)$ for some injection h, mapping v_i to x_i for each i, and every other vertices of H to vertices outside of M. This rewriting is denoted by $M \xrightarrow[R,X]{} N$. Now, this rewriting obviously extends to sets of non-terminal, for E such a set, this rewriting is denoted: $M \xrightarrow[R,E]{} N$. The *complete parallel rewriting* $\underset{R}{\Rightarrow}$ is the rewriting relative to the set of *all* non-terminal hyperarcs of R.

Now given a deterministic graph grammar $G = (N, T, R, H_0)$, and a hypergraph H, we denote by $[H] := H \cap (T\ V_H\ V_H \cup T\ V_H)$ the set of terminal arcs, and colours of H. A graph H is *generated* by G, if it belongs to the following set of isomorphic graphs:

$$G^\omega = \left\{ \cup_{n \geqslant 0} [H_n] \mid \forall n \geqslant 0, H_n \underset{R}{\Rightarrow} H_{n+1} \right\}$$

Example 2.3. We present here a simple example of deterministic graph grammar and propose a representation of the resulting graph. An important observation on this graph is that it does not provide any naming scheme for the vertices. But there is of course an obvious connection between the vertices and the sequence of graph rewriting producing them.



Graph grammars characterize regular graphs. This external characterization is very efficient to extend to these infinite graphs techniques which work for finite graphs (for example computing the connected components of a regular graph is very simple from the grammar). Furthermore these graphs correspond (in a precise sense) to transition graphs of pushdown automata. Nonetheless, algorithms which only depend on the structure of these graphs often make technical assumptions on the form of the automaton: for example that the states carry some information, such as the configuration belongs to a certain regular set. These assumptions only affect the internals of the automaton, it does not affect the structure of its configuration graph. In such case, grammars are very efficient as there is no assumption on vertices identification, only the structure is explicit.

Subsequently we will extend graph grammars in order to characterize context-sensitive languages. The way we enrich these grammars is similar to what is done for word grammars: contextual rewriting is necessary. However, in order to avoid being too general, we will have to separate context arcs from non-terminal ones. In a certain sense our characterization corresponds to do a graph rewriting *inside* a regular graph which serves as a skeleton upon which the rational graph is built.

2.3 Context-sensitive languages and rational graphs

In this section we recall the classical definition of context-sensitive languages. Then we present with some details the definition of the family of rational graphs. These graphs are very general, and provide a *graph* characterization of these languages. More details can be found in [Mor00, MS01].

Context-sensitive languages are defined as the level 1 of the Chomsky hierarchy (0 being recursively enumerable sets). Which means they are characterized by growing word grammars. Another popular characterization of these languages is by linear bounded Turing machines [Kur64].

This family of languages is very expressive, for example, the sets of words of the form ww, or $a^n b^n c^n$, with n a natural number are context-sensitive sets of words. The set of a^p where p is a prime number is context-sensitive as well. One of the most stunning property of these languages is that they are closed under complementation. These languages have countless characterizations. In this paper we will use rational graphs.

The family of rational subsets of a monoid $(M, \cdot)$ is the least family containing the finite subsets of M and closed under union, concatenation and iteration.

A transducer is a finite automaton labelled by pairs of words over a finite alphabet X, see for example [Ber79]. A transducer accepts a relation in $X^* \times X^*$; these relations are called rational relations as they are rational subsets of the product monoid $(X^* \times X^*, \cdot)$.

Now, let us consider the graphs of $X^* \times \Sigma \times X^*$. Rational graphs, denoted by $Rat(X^* \times \Sigma \times X^*)$, are extensions of rational relations, which are defined by *labelled rational transducers*.

Definition 2.4. A *labelled rational transducer* $T = (Q, I, F, E, L)$ over X and Σ, is composed of a finite set of states Q, a set of initial states $I \subseteq Q$, a set of final states $F \subseteq Q$, a finite set of transitions (or edges) $E \subseteq Q \times X^* \times X^* \times Q$ and a mapping L from F into 2^Σ.

An arc $u \xrightarrow{a} v$ is *accepted* by a labelled transducer T if there is a path from a state in I to a state f in F labelled by (u, v) and such that $a \in L(f)$.

Definition 2.5. A graph in $X^* \times \Sigma \times X^*$ is *rational* if it is accepted by a labelled rational transducer.

Let G be a rational graph, for each a in Σ we denote by G_a the restriction of G to arcs labelled by a (it defines a rational relation between vertices); let u be a vertex in X^*, we denote by $G_a(u)$ the set of all vertices v such that $u \xrightarrow{a} v$ is an arc of G.

Example 2.6. In Figure 2.1, the graph on the right-hand side is generated by the labelled transducer on the left-hand side.

The path $p \xrightarrow{0/0} q_1 \xrightarrow{0/1} r_2 \xrightarrow{1/1} r_2$ accepts the couple $(001, 011)$, the final state r_2 is labelled by b thus there is a arc $001 \xrightarrow{b} 011$ in the graph.

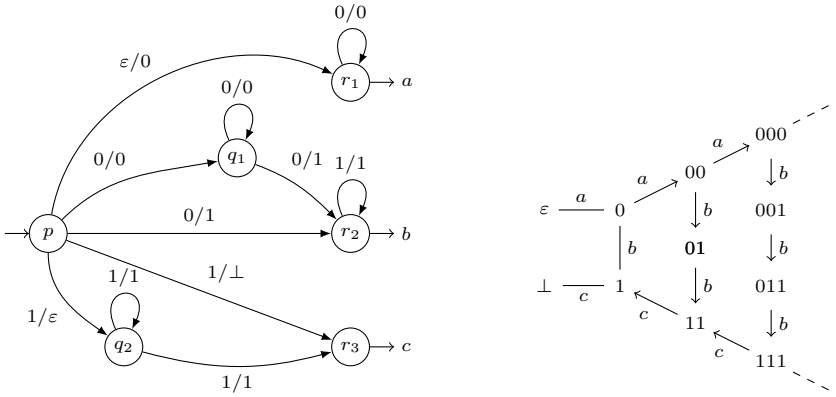


Figure 2.1: A rational graph and its labelled transducer

Rational graphs have been introduced in order to extend existing families of graphs. They provide a very general family of graphs. They have few decidable properties, but they characterize context-sensitive languages [MS01]. If we only consider trees (rooted connected acyclic-graphs such that each vertex has at most one predecessor) these trees have a decidable first order theory [CM06b].

Using transducers to characterize a family of graphs induce that each graph is defined in a very precise way. In particular, each vertex is a word, and thus each arc is defined between two precise words, which are not interchangeable. In contrast, for finite graphs, every algorithm, every characterization, is given *up to renaming of the vertices.* in fact, when dealing with infinite graphs it is difficult to avoid explicit naming of the vertices. Still, there are characterizations which overcome this problem. These characterizations are said *external*, they elude explicit naming of the vertices. Most of these characterizations rely on graph transformations unaffected by the name of the vertices.

Formally, two graphs G_1 and G_2 in $X^* \times \Sigma \times X^*$ are *isomorphic*, if there is a bijection $\psi : V(G_1) \to V(G_2)$ such that: $s_1 \xrightarrow[G_1]{d} s_2$ if and only if $\psi(s_1) \xrightarrow[G_2]{d} \psi(s_2)$.

Two isomorphic graphs have the same structure: they are the same up to a renaming of the vertices.

Finally, we recall the characterization of context-sensitive languages by rational graphs:

Theorem 2.7. [MS01] *The sets of path between regular sets of vertices of rational graphs corresponds precisely to context-sensitive languages.*

3 Contextual graph grammars

3.1 Contextual graph rewriting systems

Let N_R be a finite ranked set of *non-terminals*, and T_R a finite ranked set of *terminals*.

We propose here a natural definition of contextual graph rewriting system.

Definition 3.1 (Contextual graph rewriting system)**.** A *contextual graph rewriting system* S, is a set of rules of the form $H_c \cup f\ x_1\ \cdots\ x_{\varrho(f)} \to H_c \cup H$ where $f\ x_1\ \cdots\ x_{\varrho(f)}$ is a non-terminal hyperarc, H_c is a finite *context* graph, and H is a finite hypergraph, that can share some vertices with H_c and f. Furthermore, H_c is composed only of terminal hyperarcs, and $H_c \cup f\ x_1\ \cdots\ x_{\varrho(f)}$ forms a connected hypergraph.

Figure 3.1 illustrates a contextual rewriting rule, $\{a, b, c\}$ is the set of terminals, and the set of non-terminal is $\{\mathbf{fwd}, \mathbf{bck}\}$.

Now, given a rewriting rule $H_c \cup f\ x_1\ \cdots\ x_{\varrho(f)} \to H_c \cup H$, a rewriting step in a graph G consists in finding the non-terminal f in G, such that there is a morphism h of $H_c \cup f\ x_1\ \cdots\ x_{\varrho(f)}$ into G then removing f from G, and adding H to

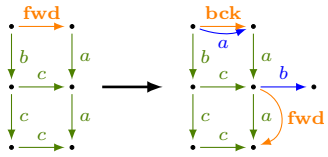


Figure 3.1: A contextual graph-rewriting rule

G according to the rule, and to the morphism h. Given a contextual graph rewriting system S, and a finite graph H, we define $S^\omega(H)$ in the same way as G^ω for a HR-grammar G. Recall that this graph is a restriction to terminal symbols, in particular in our setting it only has hyperarcs of arity 1 or 2.

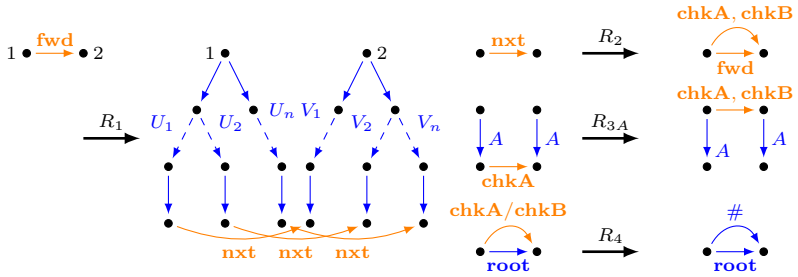
We say that such a contextual rewriting system is deterministic, if there is, at most, one rule for each non-terminal. This restriction only *limits* non-determinism, as there might be some situations where the context of a non-terminal can be found more than once. In such situation, it means that the rewriting system may generate at least two non-isomorphic graphs.

Unfortunately this natural generalization of context-free graph grammars is much too general (even restricted to systems where there is always a single morphism to match the context).

First recall a classical undecidable problem: the Post correspondence problem (PCP). An instance of PCP is a sequence of couples of words in Σ^*: $((U_i, V_i))_{i\in[n]}$, and the problem is to determine whether there is an integer k, and a sequence $(i_\ell) \in [n]^k$ such that $U_{i_1}U_{i_2}\dots U_{i_k} = V_{i_1}V_{i_2}\dots V_{i_k}$.

Proposition 3.2. *Given $(U_i, V_i)_{i\in[n]}$ an instance of PCP, there exists a graph obtained from a finite axiom A by a contextual graph rewriting system which possesses an arc labelled # between the two vertices v_0 and v_1 of A if and only if $(U_i, V_i)_{i\in[n]}$ is a positive instance.*

Example 3.3. The proof of Proposition 3.2 is in the full paper. But the construction is straightforward, and illustrated by this example. Consider $((U_i, V_i))_{i\in[n]}$ an instance of PCP, and observe the following contextual rewriting system:



The axiom is simply the following finite graph: $\{\mathbf{root}\ v_0\ v_1, \mathbf{fwd}\ v_0\ v_1\}$. Furthermore there is a rule R_{3B} similar to R_{3A} for the rewriting of **chkB**.

Now, the rule R_1 uses arc **fwd** to produce two partial binary trees corresponding to the U_i's and V_i's. For each sequence of indexes $(k_j)_{j\in[m]}$, the extremity of the path $(U_{k_j})_{j\in[m]}$ is connected to the extremity of $(V_{k_j})_{j\in[m]}$ by an non-terminal arc **nxt**. Then the rules R_{3A} and R_{3B} will ultimately reach the arc **root** if and only if $(U_i, V_i)_{i\in[n]})$ is a positive instance of PCP.

Proof. Let us first suppose, without loss of generality that this instance is over a two letters alphabet (say $X = \{A, B\}$). Our set of terminal symbols will be composed of A, B, **root** and #. Using classical encoding techniques (like $A \to AB, B \to ABB$ and $\mathbf{root} \to ABBB$) we might as well use only a two letters terminal alphabet. The set of non-terminals is formed of arity 2 labels: $\{\mathbf{fwd}, \mathbf{chkA}, \mathbf{chkB}, \mathbf{chk}\}$. The axiom A is simply the following finite graph: $\{\mathbf{root}\ v_0\ v_1, \mathbf{fwd}\ v_0\ v_1\}$. We construct two deterministic finite trees (denoted respectively by Γ_U and Γ_V) labelled on X derived respectively from the set of words $\{U_i \mid i \in [n]\}$ and $\{V_i \mid i \in [n]\}$ (each word in each set is the label of precisely one path in the corresponding tree).

Now our rewriting system is the following: the non-terminal **fwd** under no context produces from its source the tree Γ_U, and from its goal Γ_V. Then, for each $i \in [n]$, it puts 4 non-terminals $\mathbf{fwd}, \mathbf{chkA}, \mathbf{chkB}, \mathbf{chk}$ between the pair of vertices extremities of respectively U_i and V_i.

The rule for **chk** if simple: under context **root** v_a v_b, the arc **chk** v_a v_b rewrites into # v_a v_b. The rule for **chkA** (*resp.* **chkB**) is as follows: the arc **chkA** v_{s2} v_{g2} (*resp.* **chkB** v_{s2} v_{g2}), under the context: A v_{s1} v_{s2} (*resp.* B v_{s1} v_{s2}), and A v_{g1} v_{g2} (*resp.* B v_{g1} v_{g2}), rewrites into the three non terminal arcs **chkA** v_{s1} v_{g1}, **chkB** v_{s1} v_{g1}, and **chk** v_{s1} v_{g1} (roughly speaking it *moves* up along two arcs A or two arcs B).

Now, if the instance $(U_i, V_i)_{i\in[n]}$ has a solution, we may construct a sequence of rewritings producing the two corresponding trees. Then the arcs **chk** will be able to reach the roots back, and produce the arc # v_a v_b. Conversely, if there is such an arc in the graph, the arcs **chk** have reached the origin, and as these arcs are only introduced at the extremities of paths of

the form $U_{i_1}U_{i_2}\cdots U_{i_k}$ and $V_{i_1}V_{i_2}\cdots V_{i_k}$, according to the rewriting rules of **chk** we have: $U_{i_1}U_{i_2}\cdots U_{i_k} = V_{i_1}V_{i_2}\cdots V_{i_k}$ for such a sequence of indexes. □

The most direct consequence of this proposition is the following:

Corollary 3.4. *Graphs generated by deterministic contextual graph rewriting systems are not recursive.*

This result imposes to introduce tighter constraints on such a graph rewriting systems in order to characterize context-sensitive languages. We introduce such restriction in the next subsection.

3.2 Contextual hyper-edge-replacement graph grammars

In this section we present a more restrictive contextual rewriting system which will be used to characterize context-sensitive languages.

Definition 3.5. A *contextual hyper-edge-replacement hypergraph grammar* (CHR-grammar for short) is a tuple (C, N, T, R_c, H_0), where C, N and T are finite ranked alphabets of respectively contextual, non-terminal and terminal symbols; R_c is a finite set of contextual rules (for each rule $H_c \cup fx_1 \dots x_{\varrho(f)} \to H_c \cup H$, the graph H_c is formed only by arcs labelled in C, and H by arcs labelled in $T \cup N$); and H_0 is the axiom: a *deterministic* regular graph formed by arcs with labels in C, and a single non-terminal hyperarc.

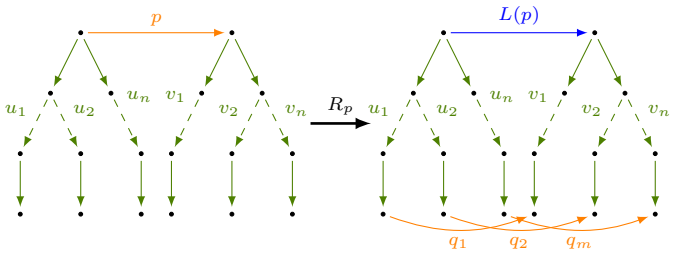
This definition imposes that the axiom is a deterministic regular graph. This restriction ensures that for each rule R, of non-terminal A, and each occurrence of A in the graph, there is at most a single morphism which maps the context of the left-hand side of R to the neighbourhood of A. This restriction may be checked, since verifying that a given graph grammar generates a deterministic graph is decidable (it a direct consequence of Proposition 3.13 in [Cau07], which states that the set of degree is finite (and computable) for a regular graph, which can be done for each label, ensuring determinism).

Later in this paper we will discuss on other structural restrictions, and indeed we will see that it is difficult to allow graphs that are not trees. And we will also have to impose stricter restrictions to the form of the rules R_c in order achieve our characterization.

First we will show that using a n-ary tree as axiom is sufficient to obtain all the rational graphs up to isomorphism, achieving the goal of containing the context-sensitive languages.

Proposition 3.6. *Any rational graph on $X^* \times \Sigma \times X^*$ is obtained from a* CHR*-grammar.*

Example 3.7. Like for Proposition 3.2, the proof is in the full paper. But the construction is straightforward, and illustrated by this example. Let G be a rational graph in $X^* \times \Sigma \times X^*$ (and T a transducer representing it), let H_0 be the complete n-ary tree labelled on X (with a non-terminal p_0 on the root). For each state p of T, we have the following rule R_p.



Here, we suppose that there are transitions $p \xrightarrow{u_i/v_i} q_i$ for some states $(q_i)_{i\in[m]}$, and also $L(p)$ represent all labels produced at state p (if p is a terminal state). Now each pair of path in H_0 correspond to a pair of paths in T. Thus the graph obtained from the contextual rewriting system is the same as the graph obtained from the transducer.

Proof. Let G be a rational graph in $X^* \times \Sigma \times X^*$, and let $T = (Q, \{q_0\}, F, E, L)$ be a labelled transducer realising it. We will use for context a graph grammar generating the complete $|X|$-ary tree labelled on X, we denote this tree by Γ_X. We define the following CHR-grammar $G_\Gamma = (X, Q, \Sigma, R_c, \Gamma_X \cup \{q_0\ v_\varepsilon\ v_\varepsilon\})$: X is the set of contexts (the labels of the Γ_X), the set of states of the transducer (Q) is the set of non-terminals, each of them is of arity 2, v_ε is the root of Γ_X, more generally, if $u \in X^*$, we will denote by v_u the vertex of Γ_X reached by the path labelled by$u$.

Now the rules of R_c copy the rules of the transducer: for each state $p \in Q$, there is a rule in R_c, the left-hand side of this rule is formed by a graph with the arc $p\ v_s\ v_g$ on top, and two finite trees (t_1 and t_2) from the vertices v_s and v_g, t_1 is the minimal deterministic tree corresponding to the paths U_i for each transition $p \xrightarrow{U_i/V_i} q_i$ in T, similarly t_2 is the tree corresponding to the V_i. Now the right-hand side does not have the arc $p\ v_s\ v_g$ anymore but there is an arc $q_i\ v_{U_i}\ v_{V_i}$ for

each q_i, and the vertices v_{U_i} and v_{V_i} are the extremities of the paths labelled respectively U_i and V_i from v_s and v_g (as Γ_X is deterministic, there is only one path for each word). Furthermore, if $p \in F$, for each $a \in L(p)$ there is an arc $a\ v_s\ v_g$. Now the CHR-grammar G_Γ puts an arc labelled a between v_u and $v_{u'}$ if and only if there is a path labelled u/u' in the transducer leading to a final state labelled a. Therefore the graph derived from the axiom, using G_Γ is precisely isomorphic to G. □

From the proof of this result, we can also add that the vertices belonging to a certain rational set can be marked by a certain terminal symbol of arity 1. Now in conjunction with Theorem 2.7 we obtain the following corollary (the construction is effective):

Corollary 3.8. *Any context-sensitive language L is the set of paths between two colours in a graph obtained from a* CHR*-grammar.*

3.3 Graphs obtained from a tree-separated contextual grammar are rational graphs

In this section we provide a converse for Proposition 3.6.

First, we designate interesting restrictions of CHR-grammar. A CHR-grammar (C, N, T, R_c, H_0) is called a *tree-*CHR*-grammar* if the axiom H_0 is a tree, and left-hand side of each rule of R_c is formed by trees *rooted* in the vertices of the non-terminal (some vertices of this non-terminal may be non-root vertices of theses trees). Furthermore, if each such tree possesses a single vertex belonging to the non-terminal (its root) this grammar is called a *tree-separated-*CHR*-grammar*. These grammars are captured by rational graphs:

Proposition 3.9. *Any graph obtained from a tree-separated-*CHR*-grammar, is isomorphic to a rational graph on $X^* \times \Sigma \times X^*$.*

There are several difficulties for establishing this result: the axiom is not a complete tree, so we need to ensure that at no moment we put an arc on a vertex that does not exist. We also need to take into account hyperarcs of arity greater than 2. And finally, we need to take into account that there may be twists: in the right-hand side of a rule, the path from the goal of the original non-terminal may lead to the source of some other non-terminal.

In order to prove Proposition 3.9 we first establish two technical lemmas.

Lemma 3.10. *Any tree-separated-*CHR*-grammar $G = (C, N, T, R_c, H_0)$, can be effectively transformed into a tree-separated-*CHR*-grammar $G' = (C, N', T, R'_c, H_0)$ generating the same set of graphs and such that each non-terminal of N' that appears in the right-hand side of rules of R'_c has at most one vertex in each subtree.*

Proof. Let $r = H_c \cup f\ x_1\ \cdots\ x_{\varrho(f)} \to H_c \cup H$ be a rule of R_c. Let $f'\ v_1\ \cdots\ v_{\varrho(f')}$ be a non-terminal hyperarc of H. For each vertex x_i of f, let us denote by Γ_i the tree rooted in x_i. Now suppose that there are two distinct vertices v_k and $v_{k'}$ belong to Γ_i for some i. Then it means that there is a vertex v_0 of Γ_i which is an ancestor of both v_k and $v_{k'}$. So we remove f' from the right-hand side of r, and we set a new non-terminal f'' in H, such that it is connected to each vertices of f' exept for v_k and $v_{k'}$, it is instead connected to v_0.

Now the rule for f'' is simply a copy of the rule of f', but f'' has one less vertex, and the context corresponding to the to vertices v_k and $v_{k'}$ is merged from their ancestors in the rule r. Indeed this process is done for the rule r simultaneously in order to remove all such multiple occurrence of vertices in the context.

This process is done in turn for each initial rule of R_c, resulting in a set R_{c1} of extra rules. Observe that each non-terminal on the left-hand side of rules of R_{c1} has at least one vertex less than its original version in R_c. Then this process is iterated for the rules of R_{c1}, and again for R_{c2}, and so on, so forth. At each step the maximal number of vertices for non-terminal is reduced by 1, ensuring the termination of the process. □

The following Lemma is just a technical necessity which is a classical result.

Lemma 3.11. *The set of paths labels leading to a colour (arc of arity 1) in a regular tree is a regular set of word.*

Proof. Let $G = (N, T, R, H_0)$ be a graph grammar generating a tree, we want to construct a regular set of paths reaching some colour c_0. From Theorem 3.12 in [Cau07] we may assume that G is complete-outside, meaning that every vertex of a left-hand side is not the goal of a vertex of the right-hand side. We furthermore assume each terminal c_0 is produced replacing some non-terminal of arity 1, C_0. As a tree is connected, we may also assume that H_0 is a single arity 0 non-terminal (let S be this terminal). Let Q be the following set:

$$Q := \{A_k \mid A \in N \wedge 0 < k \leqslant \varrho(A)\} \cup \{S\}$$

and let us define the transitions as follows

$$\Delta := \left\{ A_i \xrightarrow{u} B_j \mid A_i, B_j \in Q \wedge v_i \xRightarrow[R(A)]{u} v'_j \right\},$$

where $R(A)$ denotes the right-hand side of the rule associated to A in R, and the path between v_i and v'_j denotes that there exists a u path leading from vertex v_i of A to the vertex v'_j of B. Then using classical finite automata techniques we

may add extra states to remove each arc labelled by a word, and keep only T-labelled transitions. Let S be the initial state of this finite automaton, and let C_0 be the only final state, we have constructed a finite automaton characterizing the set of path leading to a vertex in a graph generated by G. □

We are now able to formulate the proof of Proposition 3.9.

Proof. Let H be a graph generated by a tree-separated-CHR-grammar $G = (X, N, T, R_c, \Gamma \cup \{A_0 v_1 \ \cdots v_{\varrho(A_0)}\})$, Γ is a *deterministic* tree, and the vertices of non-terminal arcs in R_c are the root of distinct trees. Without loss of generality, from Lemma 3.10, we may assume that for each rule, each non-terminal appearing in the right-hand side of the rule has at most a single vertex in each subtree. Also up to adding a few rules, we suppose that each terminal arc is produced between two vertices of the non-terminal of the left-hand side.

We will first try to characterize the set of *arcs* of graphs in G^ω, for colours (terminal hyperarcs of arity 1), it would be a similar argument, limited to a single path.

Observe that the production of each terminal arc only requires 2 paths along Γ. We will construct a first transducer $T_0 = (Q_0, I_0, F_0, E_0, L_0)$. For each non-terminal A of arity greater or equal to 2, we have a state in Q_0 for each ordered pair of vertices of A in the rule of R_c. Precisely,

$$Q_0 = \left\{A_{(a,b)} \mid A \in N \wedge (A\ v_1\ \cdots\ a\ \cdots\ b\ \cdots\ v_{\varrho(A)} \in Dom(R_c) \vee A\ v_1\ \cdots\ b\ \cdots\ a\ \cdots\ v_{\varrho(A)} \in Dom(R_c))\right\} \cup \{S_\varepsilon\}$$

with $Dom(R_c)$ representing the set of left-hand side of the rules in R_c. The idea behind the subscript (a, b) is to take into account swaps along the path leading to the production of a terminal arc, $A_{(a,b)}$ means that for the transducer, the left-hand side of the path leads to vertex a of the hyperarc A, and the right-hand side leads to the vertex b. We will elaborate on this with the description of the set of transitions.

The set I_0 has a single element: S_ε. The set of final state is:

$$F = \left\{A_{(a,b)} \mid A_{(a,b)} \in Q_0 \wedge \exists c \in T, c\ a\ b \in R_c(A)\right\}$$

where $R_c(A)$ represents the right-hand side of the rule associated to A in R_c. Following this, the labelling function associates to each state $A_{(a,b)}$ in F the set of terminal arcs $c \in T$ such that $c\ a\ b \in R_c(A)$.

Now the set of transitions is as follows:

let A and B be two non-terminals, such that $B\ v_1\ \cdots\ v_{\varrho(B)} \in R_c(A)$, for each ordered pair of vertices (a, b) of A such that there is a path leading to a pair (a', b') of vertices in B, there is a transition $A_{(a,b)} \xrightarrow{u/v} B_{(a',b')}$, where u is the path from a to a' and v the path from b to b' (from the assumption, at the beginning of the proof there is at most a single path from each vertex of A to a vertex of B). The transition from S_ε are defined in the same way: for each ordered pair (a, b) of vertices of A_0 there is a transition $S_\varepsilon \xrightarrow{u/v} A_{0(a,b)}$ where u is the path from the root to vertex a, and v the path to vertex b.

Now, let us suppose that Γ, is Γ_X the complete $|X|$-ary tree. Then the context of each rule may be satisfied, and thus, each path in T_0 corresponds to a pair of path in Γ_X. And from the definition of L_0, each such path is labelled by the correct arc.

If Γ is not the complete binary tree, from the Lemma 3.11 we have that the set of path leading to the vertices of Γ is a regular set let L_Γ be this set (and $\mathcal{A}$ its deterministic automaton). It is not sufficient to check if the final vertices belongs to L_Γ: consider for example a non-terminal arc of arity 3, a terminal arc may be produced between the vertices 1 and 2, but if the third one does not belong to Γ, this non-terminal does not exist, and thus neither does the terminal.

In order to solve this problem, we simply have to synchronize T_0 and $\mathcal{A}$. The new transducer T_1 is derived from T_0 in the following way (we will not go into the details of each state and each transition, as it is a classical construction): each state of T_1 is composed of a state of T_0, say $A_{(a,b)}$ and for each vertex of A there is a state of $\mathcal{A}$ representing the path leading to this vertex. It is important to notice that it does not matter that the state singles out only vertices a and b, the position of each vertex of A is kept in the state (we denote by $(A_{(a,b)}, q_1, \ldots, q_{\varrho(A)})$ such a state) . For each such state $(A_{(a,b)}, q_1, \ldots, q_{\varrho(A)})$ we check whether the context of $R_c(A)$ can be satisfied (from states $q_1, \ldots, q_{\varrho(A)}$), if not the state is removed. Finally the transitions of T_1 are derived from those of T_0 updating the set of states according to the path leading from each vertex of the initial arc, to the new one.

The transducer T_1 recognises a rational graph that belongs to G^ω. Proving the desired result. □

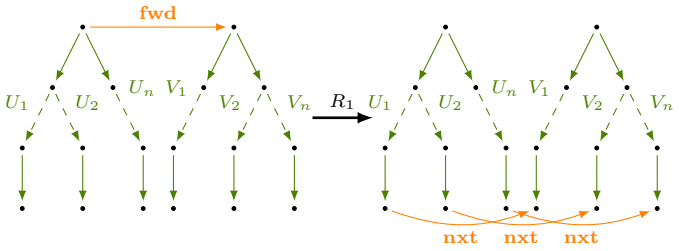
Now combining this result with Theorem 2.7 and Corollary 3.8 we obtain the desired result.

Theorem 3.12. *The set of paths (between colours) of any graph obtained from a tree-separated-CHR-grammar, is a context-sensitive language. And conversely, any context-sensitive language can be obtained as the set of paths of such a graph.*

Now we show that the natural extension of the previous result by allowing the non-terminal (of the left-hand side) to be set anywhere in the context produces another non-recursive family of graphs.

Proposition 3.13. *There is a graph obtained from a* CHR*-grammar, such that the axiom is a deterministic tree, and having a loop on the root of the axiom if and only if a given instance of PCP has a solution.*

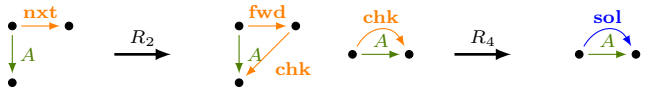
Example 3.14. Proposition 3.13 is proved using a simple modification of the proof of Proposition 3.2. We illustrate the modifications by this example. Consider $((U_i, V_i))_{i\in[n]}$ an instance of PCP, the axiom is the complete n-ary tree with a loop **fwd** on the root, and an extra terminal arc labelled **root** connecting the root to some extra vertex. The rules are the same apart from R_1:



This negative result is a consequence of some arbitrary copy moving backward the arcs. A nice solution to overcome this problem, would be to only allow a single context "above" the non-terminal. But this slight extension already induces non-recursivity of the graph. In fact, simply allowing context arcs between vertices of the non-terminal of a left-hand side is enough to obtain a non-recursive graph.

Proposition 3.15. *There is a graph obtained from a tree-*CHR*-grammar, and having a loop on the root of the axiom if and only if a given instance of PCP has a solution.*

Proof. Again, the rules are similar to the one in Proposition 3.13 but now there is no rule to return to the root of the tree. There is a simple observation: as the tree is deterministic if a sequence produces two identical words, they reach the same vertex. So the rules R_2 and R_4 are modified in the following way:



The rule R_4 may only be applied (and thus produce a **sol**-labelled arc) if and only if there is a sequence of rules leading to a single vertex. In fact we put the **sol**-arc on the vertex solution. In order to put this arc at the root of the axiom, we have to increase the arity of each non-terminal by 1 in order to leave a vertex of each non-terminal on the root. This vertex does not produce any rewriting until rule R_4 is reached, then the **sol**-loop is added on it.

□

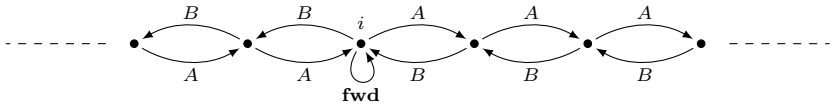
3.4 Handling context-sensitive rewriting in arbitrary regular graphs

As we have seen the situation is already quite intricate when the axiom is a tree.

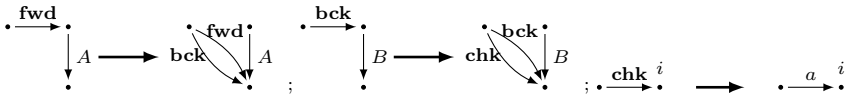
Still, some observation may prove interesting: first it is important to have a deterministic graph, ensuring that only one graph is generated up to isomorphism. As we have already seen, this is decidable for an ordinary graph grammar.

In order to consider this problem we will examine a simple example of deterministic regular graph, and a contextual CHR-grammar on this graph.

Example 3.16. Consider the following axiom:



And let us consider the graph generated from this axiom by the following contextual rewriting system (labels identifying the vertices of the context are omitted for simplicity):



This graph is very simple. Indeed it simply produces a loop on the vertex coloured i: the first rule moves to the right the non-terminal **fwd**, and produces a **bck** non-terminal which will go back to the root (using the second rule). The second rule also produces a non-terminal **chk** which produces the terminal a when it has reached the origin.

Direct application of our previous method, to this CHR-grammar, would, at some point produce the relation: $\{(\varepsilon, a^n b^n) \mid n \in \mathbb{N}\}$. First it does not correspond to the graph we are looking for, but more importantly, this relation is not rational, so there is no hope in obtaining it from a rational transducer.

The most natural way to try to solve this problem would be to *transform* the regular graph into a regular tree. And the less transforming operation would be to obtain a vertex cover of all the vertices reached from a certain colour (which is possible from [Cau07] Proposition 4.4).

Unfortunately, applying this proposition to Example 3.16 from colour i (which is the most obvious choice) results in removing the B arcs leading back to the origin (and the A arcs leading to it as well). But the transformation of the second rule would result in a rule where the context-arc B is replaced by the arc A in the other direction. Allowing such a rule would produce non recursive graphs, as stated in Proposition 3.13. Therefore this transformation is also hopeless.

4 Applications and conclusion

Applications

In this subsection we provide two direct applications and give some hints on the current lack of tools in order to manipulate the tree-separated-CHR-grammar effectively.

We first establish a result which fails for ordinary graph grammars.

Proposition 4.1. *The synchronization product of two tree-separated-*CHR*-graphs is a tree-separated-*CHR*-graph.*

sketch. The construction is straightforward: it relies on an explicit encoding of the couples of vertices. Given two graphs G_1 and G_2 derived from axioms H_1 and H_2, and sets of rules $\mathcal{R}_1$ and $\mathcal{R}_2$. Each couple (u_1, u_2), vertex of $G_1 \times G_2$, is encoded by a vertex which is reached by a path $u_1 \# u_2$ in a regular tree which is the concatenation of a copy of H_2 following an arc $\#$ from each vertex of H_1.

Now the set of rules producing $G_1 \times G_2$ corresponds to the rules of $\mathcal{R}_1$, modified to produce the initial configuration of $\mathcal{R}_2$ each time the produce a terminal arc (say a).

The modifications for the rules of $\mathcal{R}_2$ are slightly more complicated: for each non-terminal A of arity n, the rule is duplicated placing each of the vertex in either the source part of the graph or the target part: A_I, where $I \subseteq [n]$. Because the rules are tree-separated it is not difficult to keep track of these sets.

Then you produce a terminal arc a, when there is such a production, and the source of it is in the source part of the rule, and the target in the target part of the rule. □

An classical consequence of this result is the following corollary:

Corollary 4.2. *The intersection of two context-sensitive languages is a context-sensitive language.*

Using a similar technique we are able to prove the following:

Proposition 4.3. *The concatenation of two tree-separated-*CHR*-graphs is a tree-separated-*CHR*-graph. The iteration of a tree-separated-*CHR*-graph is also a tree-separated-*CHR*-graph.*

This establishes the following result:

Corollary 4.4. *The context-sensitive languages are closed under concatenation and Kleene star.*

One of the most stunning result for context-sensitive languages is closure under complementation [Sze88, Imm88]. It would be very interesting to be able to establish this result using a graph approach. Still at the moment we are not able to propose a new proof of this result because the graphs we obtain are not deterministic, and most probably, the deterministic restriction of these graphs would produce a strict sub-family.

Conclusion

In this paper we have presented a family of infinite graphs generated by contextual graph grammars. This family characterizes context-sensitive languages. We also have examined several variants which are too general and generate non-recursive graphs.

From the connection with rational graphs we are already able to state several negative decidability results for this family. But elaborating on techniques from context-free graph grammars we hope to provide algorithms for this family. In fact it

would be essential, first, to elaborate a toolbox for manipulating these graphs and then being able to provide elegant proofs of already known results.

Another interesting approach would be to identify graph rewriting systems characterizing higher order pushdown automata. Indeed the connection between deterministic graph grammars and classical pushdown automata is very natural, but there is no similar characterization for the graphs of higher order pushdown automata. Such a characterization would be a fine tool to further investigate these graphs.

More generally external characterizations are interesting for themselves, as they permit to focus on the structural properties of graphs, and to elude explicit encoding of vertices. These characterizations are the key to extend the techniques that have been used for finite graphs.

References

[Ber79] J. Berstel. *Transductions and context-free languages.* Teubner, 1979.

[BG00] A. Blumensath and E. Grädel. Automatic Structures. In *15th IEEE Symposium on Logic in Computer Science LICS 2000*, pages 51–62, 2000.

[Cau96] D. Caucal. On transition graphs having a decidable monadic theory. In Icalp 96, volume 1099 of *LNCS*, pages 194–205, 1996.

[Cau07] Didier Caucal. Deterministic graph grammars. In *Texts in logics and games 2*, pages 169–250. Amsterdam University Press, 2007.

[CK01] D. Caucal and T. Knapik. An internal presentation of regular graphs by prefix-recognizable ones. *Theory of Computing Systems*, 34(4), 2001.

[CM06a] A. Carayol and A. Meyer. Context-sensitive languages, rational graphs and determinism. *Logical Methods in Computer Science*, 2(2), 2006.

[CM06b] A. Carayol and C. Morvan. On rational trees. In Zoltán Ésik, editor, CSL 06, volume 4207 of *LNCS*, pages 225–239, 2006.

[Cou90] B. Courcelle. *Handbook of Theoretical Computer Science*, chapter Graph rewriting: an algebraic and logic approach. Elsevier, 1990.

[CW03] A. Carayol and S. Woehrle. The caucal hierarchy of infinite graphs in terms of logic and higher-order pushdown automata. In P. K. Pandya and J. Radhakrishnan, editors, FSTTCS 03, volume 2914 of *LNCS*, pages 112–123, 2003.

[Imm88] Neil Immerman. Nondeterministic space is closed under complementation. *SIAM Journal on computing*, 17(5):935–938, October 1988.

[KP99] T. Knapik and E. Payet. Synchronization product of linear bounded machines. In *FCT*, volume 1684 of *LNCS*, pages 362–373, 1999.

[Kur64] S.-Y. Kuroda. Classes of languages and linear-bounded automata. *Information and Control*, 7(2):207–223, June 1964.

[Mor00] C. Morvan. On rational graphs. In J. Tiuryn, editor, Fossacs 00, volume 1784 of *LNCS*, pages 252–266, 2000.

[MS85] D. Muller and P. Schupp. The theory of ends, pushdown automata, and second-order logic. *Theoretical Computer Science*, 37:51–75, 1985.

[MS01] C. Morvan and C. Stirling. Rational graphs trace context-sensitive languages. In A. Pultr and J. Sgall, editors, *MFCS*, volume 2136 of *LNCS*, pages 548–559, 2001.

[Ris03] C. Rispal. The synchronized graphs trace the context-sensitive languages. In *ENTCS*, volume 68, 2003.

[Sze88] R. Szelepcsényi. The method of forced enumeration for nondeterministic automata. *Acta Informatica*, 26(3):279–284, November 1988.